

## Grupo 5 - Lenguaje SQL - 2024

### TP FINAL

Nombre y apellido	Email
Ayelén Arias	aaarias@estudiantes.unsam.edu.ar
Nicole Seinhart	nseinhart@unsam.edu.ar
Olivia Zantleifer	ozantleifer@estudiantes.unsam.edu.ar
Maria Belen Quiñones	mquinones@estudiantes.unsam.edu.ar
Julian Rolando	jdrolando@estudiantes.unsam.edu.ar
Santiago Mosoni	santimosoni@gmail.com
Fabiana Fulgenzi	ffulgenzi@iib.unsam.edu.ar
Agustin Lerer	Lerer918@gmail.com

#### Presentación del motor de base de datos:

##### Empresa/organización que lo provee, libre/pago, etc.

Los motores de bases de datos son fundamentales en el mundo de la tecnología de la información, ya que proporcionan la infraestructura necesaria para almacenar, organizar, recuperar y manipular datos de manera eficiente. Para nuestro trabajo elegimos MySQL que es uno de los motores más utilizados y populares ya que es de descarga gratuita y open source. lo cual hace que pueda encontrarse fácilmente información para su uso.



El motor de trabajo de MySQL consta de varios componentes, incluyendo el servidor MySQL, el motor de almacenamiento, el optimizador de consultas y el planificador de ejecución.

El servidor MySQL actúa como el punto de entrada para las conexiones de clientes y coordina todas las operaciones de base de datos. El optimizador de consultas y el planificador de ejecución trabajan juntos para analizar y ejecutar consultas de la manera más eficiente posible, utilizando índices, estadísticas y otras técnicas de optimización.

Puede ser utilizado para varios sistemas operativos: Linux, Windows, macOS.

Por su escalabilidad, puede ser utilizado tanto para bases pequeñas como grandes sistemas.

## Cómo conseguirlo, dónde descargarlo.

Al tener una licencia de código abierto puede descargarse la página oficial, donde puede elegirse cual es el sistema operativo y compatibilidad con la máquina del usuario:

<https://www.mysql.com/>

### MySQL Community Downloads

MySQL Installer

The screenshot shows the MySQL Community Downloads page for the MySQL Installer 8.0.37. It features a navigation bar with 'General Availability (GA) Releases' and 'Archives' tabs. A note states that MySQL 8.0 is the final series with the MySQL Installer, and users should use MSI or Zip archives for MySQL 8.1 and higher. Below the note, there are dropdown menus for 'Select Version:' (set to 8.0.37) and 'Select Operating System:' (set to Microsoft Windows). A table lists two download options for Windows (x86, 32-bit), both MSI Installers. The first is the 'web-community' version (2.1M) and the second is the 'community' version (296.1M). Each entry includes an MD5 checksum and a link to the signature. A footer note suggests using MD5 checksums and GnuPG signatures to verify package integrity.

Platform	Version	Size	Download Link
Windows (x86, 32-bit), MSI Installer	8.0.37	2.1M	<a href="#">Download</a>
<small>(mysql-installer-web-community-8.0.37.0.msi)</small>			
<small>MD5: 398f1365f2bd43af9f6ece9add565c1b   <a href="#">Signature</a></small>			
Windows (x86, 32-bit), MSI Installer	8.0.37	296.1M	<a href="#">Download</a>
<small>(mysql-installer-community-8.0.37.0.msi)</small>			
<small>MD5: ae685e4f62aaf8bb1adef684d62a49f2   <a href="#">Signature</a></small>			

## Instalación & configuración para su uso.

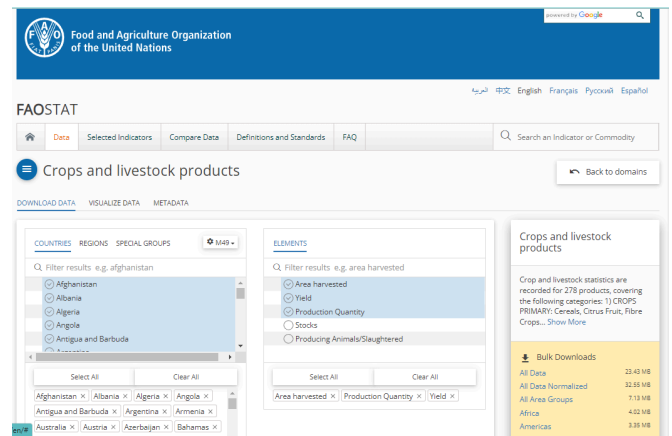
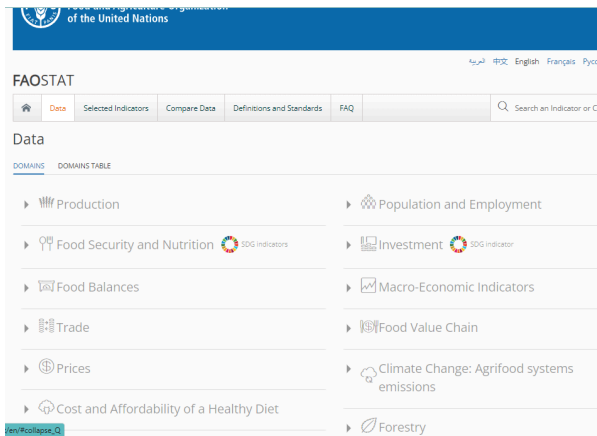
Algunas cuestiones importantes para su instalación es que debe descargarse el paquete completo que incluye MySQL y el MSI Installer.

Durante la instalación se pueden editar algunas funciones de arranque, es decir que si queremos que el servidor esté conectado siempre que encendamos la PC o no, por ejemplo.

## Elección de una Base de datos

Se decidió utilizar datos existentes es un dataset de la FAO.

Se realiza la descarga de los datos de la FAO de producción agrícola ganadera completa ("bulk download" "All data Normalized").



La tabla principal tiene los siguientes datos:

Area Code, Area Code (M49), Area, Item Code, Item Code (CPC), Item, Element Code, Element, Year Code, Year, Unit, Value, Flag

**Área:** es el país hay datos de todo el mundo

**Item:** el cultivo o tipo de producción (trigo, maíz, carne de búfalo leche, etc)

**Element:** el tipo de parámetro (rendimiento / Área / producción, etc)

**Flag:** refiere del tipo de dato (Oficial, estimado, imputed Missing or not applicable, de organización internacional)

**year:** datos desde 1961 a 2022

Hay otras tablas accesorias con la codificación de países regiones Items elementos Flag.

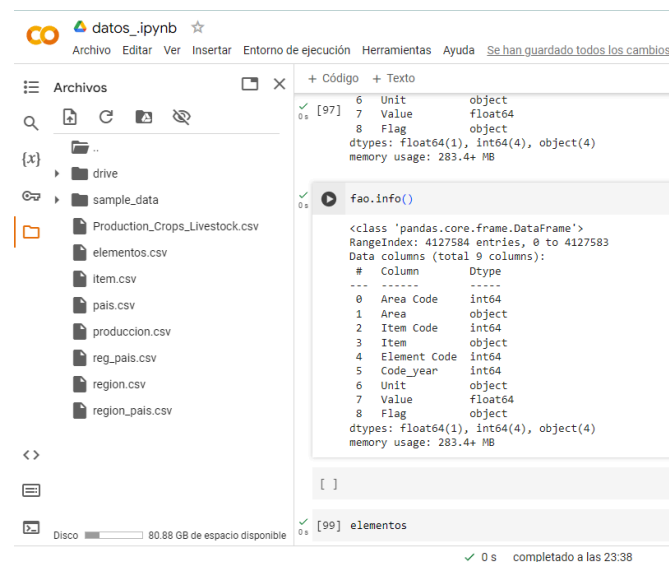
Como la tabla principal tiene 4.127.584 registros solo pudimos trabajarlos con python a través de google colab ,dado que no se podía abrir con el excel ni importar directamente en sql Workbench en tiempos razonables.

## Preparación de los datos

En este sentido, se realizó un filtrado de los países de América y del elemento producción quedándonos con 207775 registros, (posteriormente se fueron agregando más datos para la evaluación de rendimiento).

Se borraron algunas columnas para normalizar la base de datos y también se han renombrado columnas para evitar problemas con palabras reservadas como Year, y se reemplazaron los espacios por "\_" y en otros casos simplemente para hacerlos más intuitivos. La tabla de países incluía, las regiones y continentes, por lo que esta tabla se dividió en tres distintas una para cada uno de los "niveles".

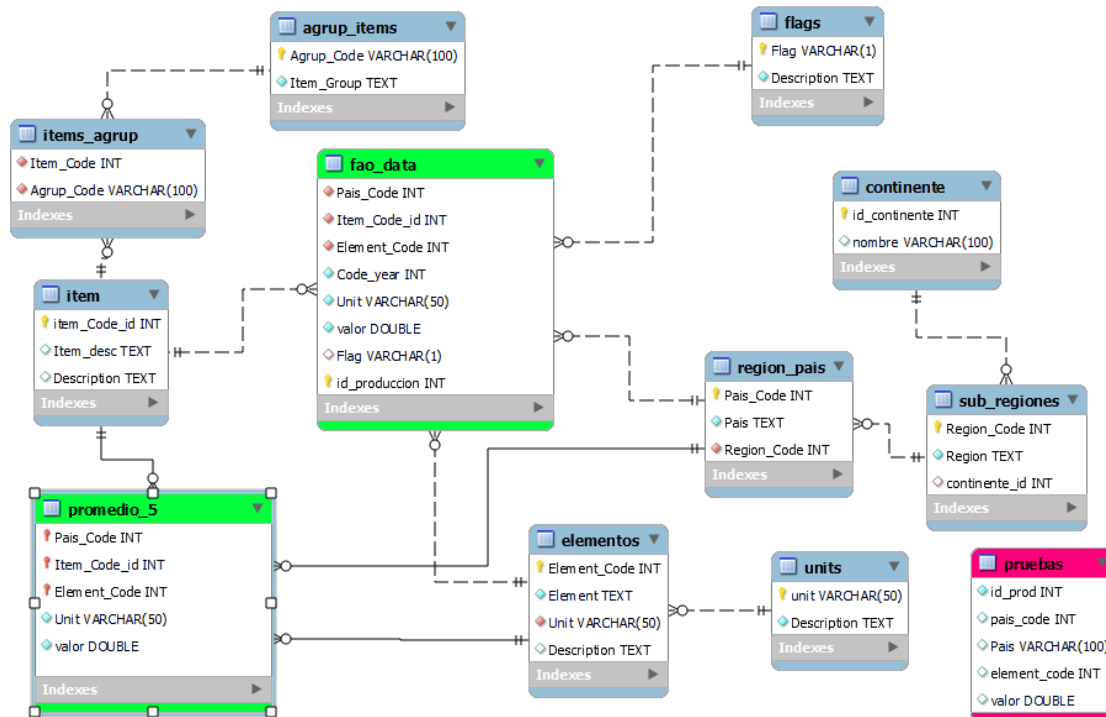
De manera análoga la tabla de ítems también tenía códigos para la agrupación de las distintas producciones; en este caso solo nos quedamos con las producciones y el primer nivel de agrupación quedando una relación muchos a muchos entre *agrup\_items* e *item* por lo que se genera la tabla intermedia.



## Modelo final de la base de datos:

(Archivos 00\_DER\_g5\_fao\_der\_V3.mwb y 01\_SCHEMS\_G5\_fao\_der\_V3.sql)

Se generaron tablas adicionales para el promedio de 5 años y otra de Pruebas para evaluación de rendimiento.



Las tablas armadas con python fueron importadas a MySQL con “data import wizard” y, se generaron las relaciones, se establecieron las PK y FK, y también hubo que ajustar algunos tipos de datos por ejemplo SQL no permite usar como PK un campo tipo “text” cambiándose a varchar()

## Requerimientos:

(Archivos 1\_TP\_FINAL\_G5\_Operaciones\_Generales.sql y 2\_TP\_FINAL\_G5\_Queries.sql)

### Operaciones sobre una tabla:

#### Creación

```
CREATE TABLE IF NOT EXISTS `g5_fao_der`.`sub regiones` (  
  `Region_Code` INT NOT NULL,  
  `Region` TEXT NOT NULL,  
  `continente_id` INT NULL DEFAULT NULL,  
  PRIMARY KEY (`Region_Code`),  
  CONSTRAINT `fk_continente`  
    FOREIGN KEY (`continente_id`)  
    REFERENCES `g5_fao_der`.`continente` (`id_continente`))  
ENGINE = InnoDB  
DEFAULT CHARACTER SET = utf8mb4  
COLLATE = utf8mb4_0900_ai_ci;
```

## Eliminación

```
DELETE FROM `fao_data` WHERE  
Pais_Code = 109 and item_Code_id = 572 and  
Element_Code = 5510 and Code_Year = 1961;
```

Pais_Code	Item_Code_id	Element_Code	Code_year	Unit	valor	Flag	id_produccion
109	572	5510	1961	t	2887	A	1
109	486	5510	1961	t	176...	A	63
109	176	5510	1961	t	22	X	125
109	51	5510	1961	t	17766	A	187
109	1183	5510	1961	t	122	E	248
109	181	5510	1961	t	45	X	310
109	358	5510	1961	t	6096	A	379
109	568	5510	1961	t	0	M	441
109	426	5510	1961	t	758	A	503
109	125	5510	1961	t	9070	A	565
109	869	5510	1961	t	312	I	627
109	393	5510	1961	t	202	A	689
109	689	5510	1961	t	2200	E	751
109	401	5510	1961	t	610	A	813
109	661	5510	1961	t	2870	A	875
109	252	5510	1961	t	811...	I	937

Pais_Code	Item_Code_id	Element_Code	Code_year	Unit	valor	Flag	id_produccion
109	486	5510	1961	t	176...	A	63
109	176	5510	1961	t	22	X	125
109	51	5510	1961	t	17766	A	187
109	1183	5510	1961	t	122	E	248
109	181	5510	1961	t	45	X	310
109	358	5510	1961	t	6096	A	379
109	568	5510	1961	t	0	M	441
109	426	5510	1961	t	758	A	503
109	125	5510	1961	t	9070	A	565
109	869	5510	1961	t	312	I	627
109	393	5510	1961	t	202	A	689
109	689	5510	1961	t	2200	E	751
109	401	5510	1961	t	610	A	813
109	661	5510	1961	t	2870	A	875

## Inserción

```
INSERT INTO `fao_data` VALUES  
(109,572,5510,1962,'t',2315,'A',2);
```

Pais_Code	Item_Code_id	Element_Code	Code_year	Unit	valor	Flag	id_produccion
109	572	5510	1961	t	2887	A	1
109	572	5510	1963	t	2315	A	3
109	572	5510	1964	t	2588	A	4
109	572	5510	1965	t	2452	A	5
109	572	5510	1966	t	2588	A	6
109	572	5510	1967	t	2452	A	7
109	572	5510	1968	t	2043	A	8
109	572	5510	1969	t	2315	A	9
109	572	5510	1970	t	2452	A	10
109	572	5510	1971	t	2452	A	11

Pais_Code	Item_Code_id	Element_Code	Code_year	Unit	valor	Flag	id_produccion
109	572	5510	1961	t	2887	A	1
109	572	5510	1962	t	2315	A	2
109	572	5510	1963	t	2315	A	3
109	572	5510	1964	t	2588	A	4
109	572	5510	1965	t	2452	A	5
109	572	5510	1966	t	2588	A	6
109	572	5510	1967	t	2452	A	7
109	572	5510	1968	t	2043	A	8
109	572	5510	1969	t	2315	A	9
109	572	5510	1970	t	2452	A	10

## Actualización

cambia Argentina por Arg en el campo País

```
select * from region_pais group by Pais;  
UPDATE region_pais SET Pais = "Arg"  
WHERE Pais = "Argentina";  
(requiere SET SQL_SAFE_UPDATES = 0);
```

Pais_Code	Pais	Region_Code
2	Afghanistan	5303
3	Albania	5403
8	Antigua and Barbuda	5206
9	Argentina	5207
1	Armenia	5305
10	Australia	5501
11	Austria	5404
52	Azerbaijan	5305
12	Bahamas	5206
13	Bahrain	5305

Pais_Code	Pais	Region_Code
2	Afghanistan	5303
3	Albania	5403
8	Antigua and Barbuda	5206
9	Arg	5207
1	Armenia	5305
10	Australia	5501
11	Austria	5404
52	Azerbaijan	5305
12	Bahamas	5206
13	Bahrain	5305

## Índices

```
34 # Creación de índice para Nombre de continentes
35 • ALTER TABLE `g5_fao_der`.`continente`
36 ADD INDEX `continente` (`nombre` ASC) VISIBLE;
37 ;
38 • SHOW INDEXES FROM continente;
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible
continente	0	PRIMARY	1	id_continente	A	5	NULL	NULL		BTREE			YES
continente	1	continente	1	nombre	A	5	NULL	NULL	YES	BTREE			YES

```
40
41 • SHOW INDEXES FROM continente;
42
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible
continente	0	PRIMARY	1	id_continente	A	5	NULL	NULL		BTREE			YES

```
# uso de índice único para establecer la restricción necesaria para no cargar 2 veces un mismo dato
• ALTER TABLE `g5_fao_der`.`fao_data`
  ADD UNIQUE INDEX `restrict` (`Pais_Code` ASC, `Item_Code_id` ASC, `Element_Code` ASC,
    `Code year` ASC, `Unit` ASC, `Flag` ASC) VISIBLE,
```

## Búsquedas de una clave y de 2 claves

```
1 # Búsqueda por clave país!!! ;)
2 • SELECT * FROM region_pais
3 WHERE region_pais.Pais_Code = 9 or region_pais.Pais_Code = 68;
4 |
5
```

	Pais_Code	Pais	Region_Code
9		Argentina	5207
68		France	5404
NULL	NULL	NULL	NULL

```
5
6 #Búsqueda una tabla por 2 claves (en realidad usamos 3 FK dada la cantidad de datos, item 15 trigo)
7 • Select * from fao_data
8 where Pais_Code =9 and Item_Code_id =15 and Code_year=2022;
9
```

[illegible]

**Repetir las operaciones para 3 tablas generando una relación con el siguiente aspecto:**

Se van a realizar sobre las siguientes tablas:

**continente → sub\_regiones → region\_pais**

Realizar consultas para la obtención de datos de la siguiente forma:

**A partir de un dato contenido en A conseguir uno de C**

Queries to final\* region\_pais elementos TP\_FINAL\_G5\_SP\_FUNCNT\* promedio\_5

```
9 # desde continente a paises (países de America)
10 • SELECT continente.nombre as Continente, sub_regiones.Region, region_pais.Pais_Code, region_pais.Pais FROM region_pais
11 INNER JOIN sub_regiones ON region_pais.Region_Code = sub_regiones.Region_Code
12 INNER JOIN continente ON sub_regiones.continente_id = continente.id_continente
13 WHERE continente.nombre = 'America'
14 ORDER BY region_pais.Pais;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

Continente	Region	Pais_Code	Pais
America	Caribbean	8	Antigua and Barbuda
America	South America	9	Argentina
America	Caribbean	12	Bahamas
America	Caribbean	14	Barbados
America	Central America	23	Belize
America	South America	19	Bolivia (Plurinational State of)
America	South America	21	Brazil
America	Northern America	33	Canada
America	South America	40	Chile
America	South America	44	Colombia
America	Central America	48	Costa Rica
America	Caribbean	49	Cuba
America	Caribbean	55	Dominica
America	Caribbean	56	Dominican Republic
America	South America	58	Ecuador
America	Central America	60	El Salvador

Result 70 x

**A partir de un dato contenido en C conseguir uno de B desde país → sub\_región**

```
16
17 # desde pais a sub_region, todos los países que empiezan por A
18 • SELECT sub_regiones.Region, region_pais.Pais FROM region_pais
19 INNER JOIN sub_regiones ON region_pais.Region_Code = sub_regiones.Region_Code
20 WHERE region_pais.Pais LIKE "A%"
21 ORDER BY sub_regiones.Region, region_pais.Pais;
22
23
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

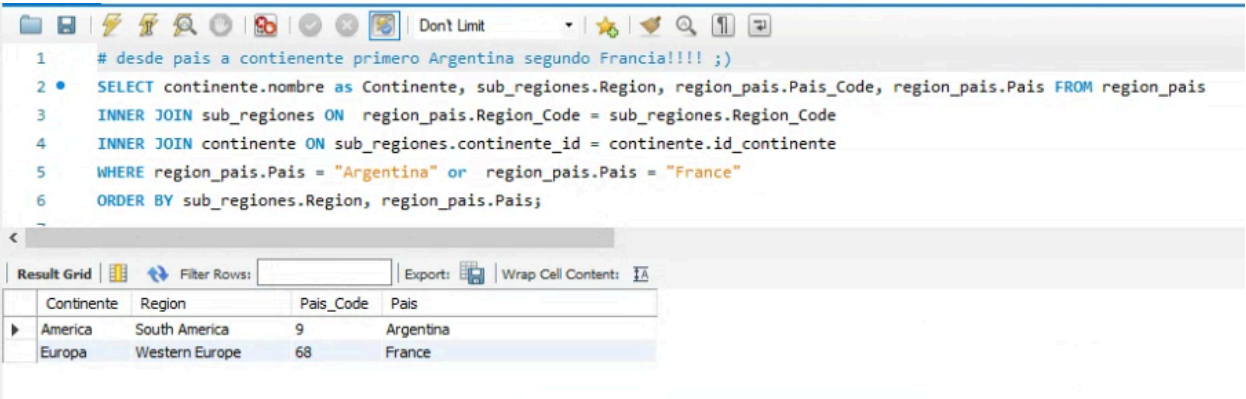
Region	Pais
Australia and New Zealand	Australia
Caribbean	Antigua and Barbuda
South America	Argentina
Southern Asia	Afghanistan
Southern Europe	Albania
Western Asia	Armenia
Western Asia	Azerbaijan
Western Europe	Austria

Result 34 x

**A partir de un dato contenido en C conseguir uno de A**



desde País → sub\_región → Continente



```

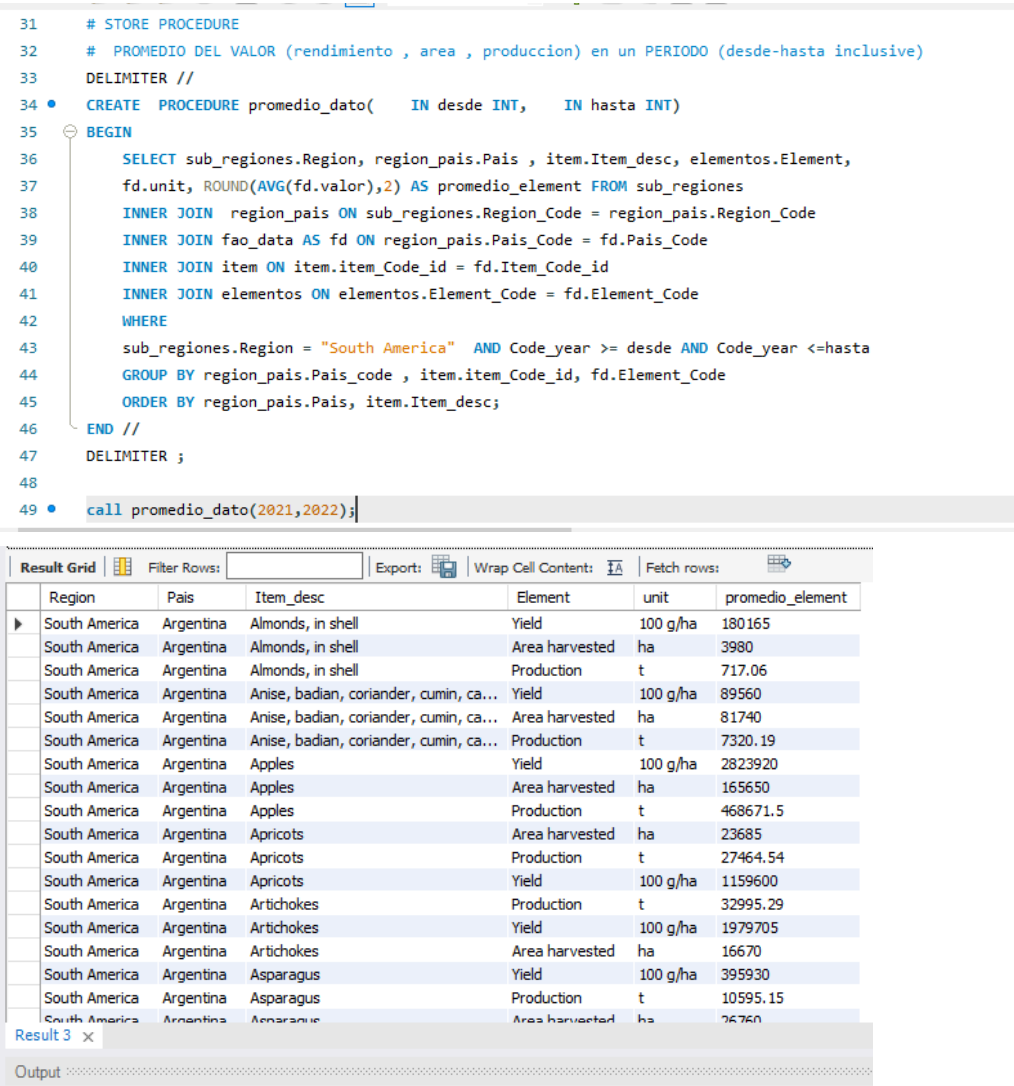
1 # desde pais a contienente primero Argentina segundo Francia!!!! ;)
2 • SELECT continente.nombre as Continente, sub_regiones.Region, region_pais.Pais_Code, region_pais.Pais FROM region_pais
3 INNER JOIN sub_regiones ON region_pais.Region_Code = sub_regiones.Region_Code
4 INNER JOIN continente ON sub_regiones.continente_id = continente.id_continente
5 WHERE region_pais.Pais = "Argentina" or region_pais.Pais = "France"
6 ORDER BY sub_regiones.Region, region_pais.Pais;

```

Continente	Region	Pais_Code	Pais
America	South America	9	Argentina
Europa	Western Europe	68	France

## Realizar Un SP, 1 Function y 1 Trigger y ejemplificar su uso (Archivo 3\_TP\_FINAL\_G5\_SP\_FUN\_TRIG.sql)

**SP:** calcular promedio de los distintos “elementos” (superficie, rendimiento o producción) para los países se “South América” para el periodo indicado.



```

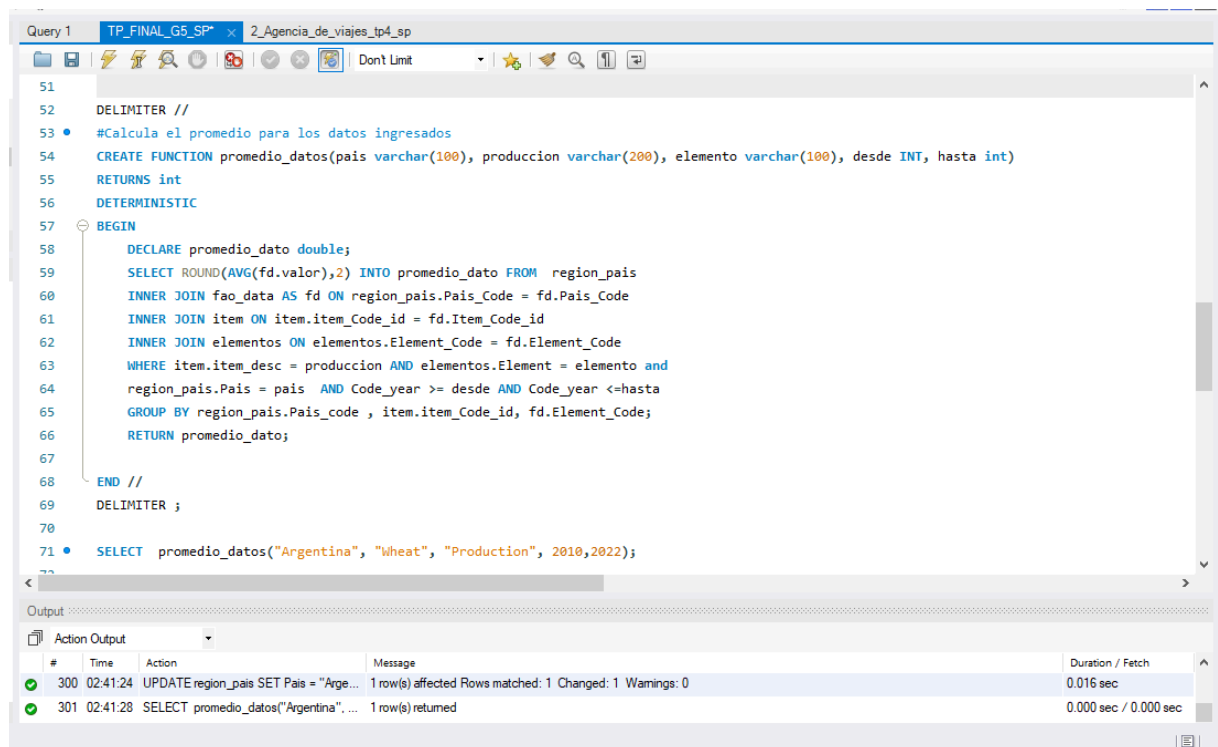
31 # STORE PROCEDURE
32 # PROMEDIO DEL VALOR (rendimiento , area , produccion) en un PERIODO (desde-hasta inclusive)
33 DELIMITER //
34 • CREATE PROCEDURE promedio_dato( IN desde INT, IN hasta INT)
35 BEGIN
36 SELECT sub_regiones.Region, region_pais.Pais , item.Item_desc, elementos.Element,
37 fd.unit, ROUND(AVG(fd.valor),2) AS promedio_element FROM sub_regiones
38 INNER JOIN region_pais ON sub_regiones.Region_Code = region_pais.Region_Code
39 INNER JOIN fao_data AS fd ON region_pais.Pais_Code = fd.Pais_Code
40 INNER JOIN item ON item.item_Code_id = fd.Item_Code_id
41 INNER JOIN elementos ON elementos.Element_Code = fd.Element_Code
42 WHERE
43 sub_regiones.Region = "South America" AND Code_year >= desde AND Code_year <= hasta
44 GROUP BY region_pais.Pais_code , item.item_Code_id, fd.Element_Code
45 ORDER BY region_pais.Pais, item.Item_desc;
46 END //
47 DELIMITER ;
48
49 • call promedio_dato(2021,2022);

```

Region	Pais	Item_desc	Element	unit	promedio_element
South America	Argentina	Almonds, in shell	Yield	100 g/ha	180165
South America	Argentina	Almonds, in shell	Area harvested	ha	3980
South America	Argentina	Almonds, in shell	Production	t	717.06
South America	Argentina	Anise, badian, coriander, cumin, ca...	Yield	100 g/ha	89560
South America	Argentina	Anise, badian, coriander, cumin, ca...	Area harvested	ha	81740
South America	Argentina	Anise, badian, coriander, cumin, ca...	Production	t	7320.19
South America	Argentina	Apples	Yield	100 g/ha	2823920
South America	Argentina	Apples	Area harvested	ha	165650
South America	Argentina	Apples	Production	t	468671.5
South America	Argentina	Apricots	Area harvested	ha	23685
South America	Argentina	Apricots	Production	t	27464.54
South America	Argentina	Apricots	Yield	100 g/ha	1159600
South America	Argentina	Artichokes	Production	t	32995.29
South America	Argentina	Artichokes	Yield	100 g/ha	1979705
South America	Argentina	Artichokes	Area harvested	ha	16670
South America	Argentina	Asparagus	Yield	100 g/ha	395930
South America	Argentina	Asparagus	Production	t	10595.15
South America	Argentina	Asparagus	Area harvested	ha	76760



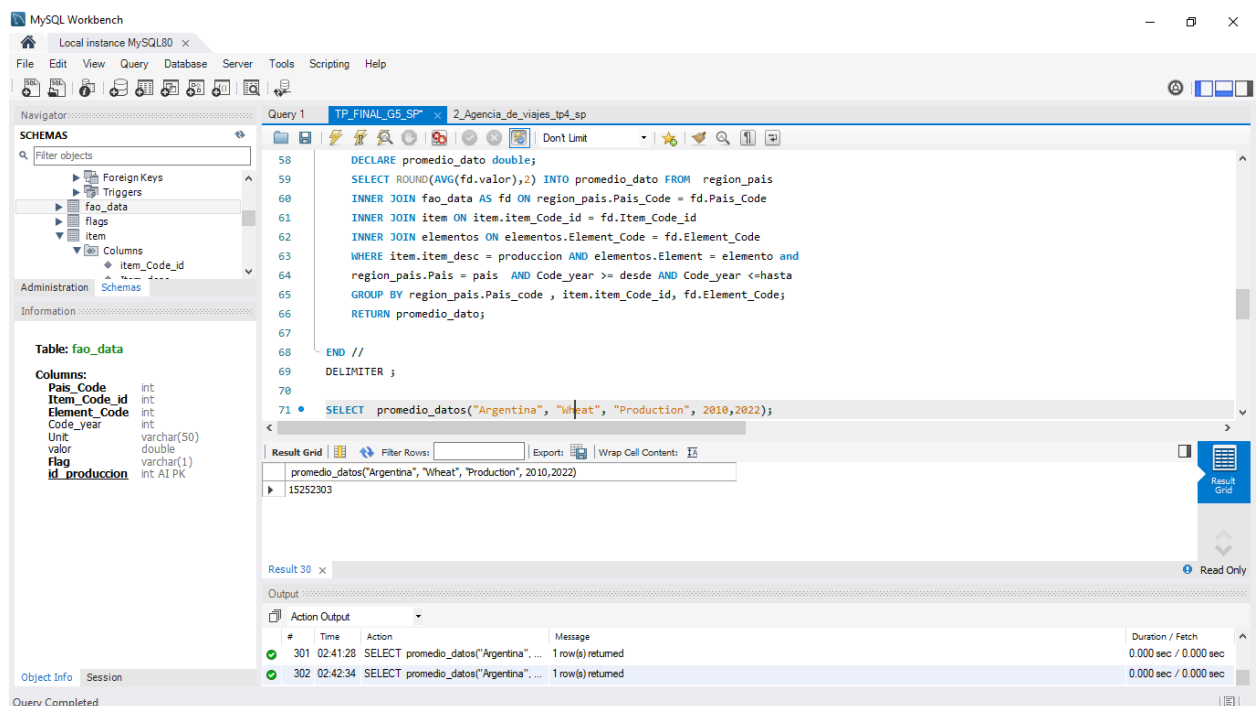
**Function:** promedio de x últimos años (desde-hasta inclusive) x producto x pais x elemento



The screenshot shows the MySQL Workbench interface with a query window titled 'Query 1' containing a stored function definition and its execution. The function, named 'promedio\_datos', calculates the average production value for a specific country, product, and element over a range of years. The execution output shows two actions: an update to the 'region\_pais' table and a successful call to the 'promedio\_datos' function for Argentina, Wheat, and Production in 2010 and 2022.

```
51
52 DELIMITER //
53 • #Calcula el promedio para los datos ingresados
54 CREATE FUNCTION promedio_datos(pais varchar(100), produccion varchar(200), elemento varchar(100), desde INT, hasta int)
55 RETURNS int
56 DETERMINISTIC
57 BEGIN
58     DECLARE promedio_datos double;
59     SELECT ROUND(AVG(fd.valor),2) INTO promedio_datos FROM region_pais
60     INNER JOIN fao_data AS fd ON region_pais.Pais_Code = fd.Pais_Code
61     INNER JOIN item ON item.item_Code_id = fd.Item_Code_id
62     INNER JOIN elementos ON elementos.Element_Code = fd.Element_Code
63     WHERE item.item_desc = produccion AND elementos.Element = elemento and
64     region_pais.Pais = pais AND Code_year >= desde AND Code_year <= hasta
65     GROUP BY region_pais.Pais_code , item.item_Code_id, fd.Element_Code;
66     RETURN promedio_datos;
67
68 END //
69 DELIMITER ;
70
71 • SELECT promedio_datos("Argentina", "Wheat", "Production", 2010,2022);
```

#	Time	Action	Message	Duration / Fetch
✓ 300	02:41:24	UPDATE region_pais SET Pais = "Arge...	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0	0.016 sec
✓ 301	02:41:28	SELECT promedio_datos("Argentina", ...	1 row(s) returned	0.000 sec / 0.000 sec



This screenshot provides a more detailed view of the MySQL Workbench interface. On the left, the 'SCHEMAS' panel shows the structure of the 'fao\_data' table, including columns like Pais\_Code, Item\_Code\_id, Element\_Code, Code\_year, Unit, valor, Flag, and id\_production. The main query window shows the same function definition as the previous screenshot. The 'Result Grid' at the bottom displays the output of the function call, showing a single row with the value 15252303. The 'Output' panel at the very bottom shows the execution log for two queries, both returning one row.

**Table: fao\_data**

Column	Type
Pais_Code	int
Item_Code_id	int
Element_Code	int
Code_year	int
Unit	varchar(50)
valor	double
Flag	varchar(1)
id_production	int AI PK

```
58 DECLARE promedio_datos double;
59 SELECT ROUND(AVG(fd.valor),2) INTO promedio_datos FROM region_pais
60 INNER JOIN fao_data AS fd ON region_pais.Pais_Code = fd.Pais_Code
61 INNER JOIN item ON item.item_Code_id = fd.Item_Code_id
62 INNER JOIN elementos ON elementos.Element_Code = fd.Element_Code
63 WHERE item.item_desc = produccion AND elementos.Element = elemento and
64 region_pais.Pais = pais AND Code_year >= desde AND Code_year <= hasta
65 GROUP BY region_pais.Pais_code , item.item_Code_id, fd.Element_Code;
66 RETURN promedio_datos;
67
68 END //
69 DELIMITER ;
70
71 • SELECT promedio_datos("Argentina", "Wheat", "Production", 2010,2022);
```

Column	Value
promedio_datos("Argentina", "Wheat", "Production", 2010,2022)	15252303

#	Time	Action	Message	Duration / Fetch
✓ 301	02:41:28	SELECT promedio_datos("Argentina", ...	1 row(s) returned	0.000 sec / 0.000 sec
✓ 302	02:42:34	SELECT promedio_datos("Argentina", ...	1 row(s) returned	0.000 sec / 0.000 sec

**Trigger:** Informa si se intenta ingresar datos para un país inexistente en la tabla de países (region\_pais)

```

54 DELIMITER //
55 • CREATE TRIGGER pais_inexistente_trigger
56 BEFORE INSERT ON fao_data
57 FOR EACH ROW
58 BEGIN
59     DECLARE pais_existente INT;
60     DECLARE mensaje VARCHAR(255);
61     -- Verificar si el nuevo Pais_Code existe en la tabla region_pais
62     SELECT COUNT(*) INTO pais_existente FROM region_pais WHERE Pais_Code = NEW.Pais_Code;
63     -- Si el Pais_Code no existe, generar un mensaje de error
64     IF pais_existente = 0 THEN
65         SET mensaje = CONCAT('Pais_code ', NEW.Pais_Code, ' no existente en la tabla region_pais');
66         SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = mensaje;
67     END IF;
68 END//
69 DELIMITER ;
70 • INSERT INTO fao_data (Pais_Code, Item_Code_id, Element_Code, Code_year, Unit, valor, Flag)
71 VALUES ('569', '572', '6000', '2024', 't', '1000', 'A');

```

Output					
Action Output					
#	Time	Action	Message	Duration / Fetch	
1	21:28:14	INSERT INTO fao_data (Pais_Code, Item_Code_id, Element_Code, Code_year, Unit, val...	Error Code: 1644. Pais_code 569 no existente en la tabla region_pais	0.016 sec	

**Trigger 2:** Uno de los requerimientos estipulados fue mantener actualizada una tabla de promedio de los últimos 5 años para cada dato/pais/item/elemento

Queries tp final* region_pais elementos TP_FINAL_G5_SP_FU					
<pre> 1 • SELECT * FROM g5_fao_der.promedio_5 2   where Item_Code_id = 15 and Pais_Code = 9; </pre>					
Result Grid					
	Pais_Code	Item_Code_id	Element_Code	promedio	Unit
	9	15	5312	63091086	ha
	9	15	5419	309546	100 g/ha
	9	15	5510	19509855.6	t

```

75 • CREATE TRIGGER actualizar_promedio_5
76 after INSERT ON fao_data
77 FOR EACH ROW
78 BEGIN
79     DECLARE promedio Double;
80     -- Calcular nuevo promedio
81     SELECT Round(Avg(valor),2) INTO promedio FROM fao_data WHERE Pais_Code = NEW.Pais_Code
82     and Item_Code_id = new.Item_Code_id and Element_Code = new.Element_Code and
83     Code_year > (new.Code_year-5) and Unit = new.Unit;
84     -- Actualizo el nuevo promedio
85     SET SQL_SAFE_UPDATES = 0;
86     REPLACE INTO promedio_5 (Pais_Code, Item_Code_id, Element_Code, promedio, Unit)
87     VALUES(NEW.Pais_Code, new.Item_Code_id, new.Element_Code, promedio, new.Unit);
88     SET SQL_SAFE_UPDATES = 1;
89
90 END//
91 DELIMITER ;
92
93 • INSERT INTO fao_data (Pais_Code, Item_Code_id, Element_Code, Code_year, Unit, valor, Flag)
94 VALUES (9, 15, 5419, 2023, '100 g/ha',1, 'A');
95 • INSERT INTO fao_data (Pais_Code, Item_Code_id, Element_Code, Code_year, Unit, valor, Flag)
96 VALUES (9, 15, 5419, 2024, '100 g/ha',1, 'A');
97 • INSERT INTO fao_data (Pais_Code, Item_Code_id, Element_Code, Code_year, Unit, valor, Flag)
98 VALUES (9, 15, 5419, 2025, '100 g/ha',1, 'A');

```

Queries tp final* region_pais elementos TP_FINAL_G5_SP_FUNCNT*					
<pre> 1 • SELECT * FROM g5_fao_der.promedio_5 2   where Item_Code_id = 15 and Pais_Code = 9; </pre>					
Result Grid					
	Pais_Code	Item_Code_id	Element_Code	promedio	Unit
	9	15	5312	63091086	ha
	9	15	5419	122840.6	100 g/ha
	9	15	5510	19509855.6	t
*	NULL	NULL	NULL	NULL	NULL

## Análisis de performance:

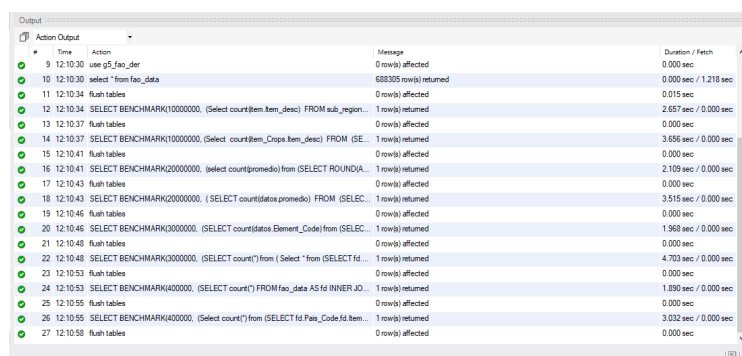
Análisis de performance de consultas consideradas óptimas contra otras consideradas “subóptimas”, (con subconsultas ordenamiento y/o agrupamiento por campos no indexados )  
Se tomó la precaución de usar “flush tables” previo a cada consulta.

Estas corridas son con 688.305 registros en la tabla principal

### Archivo 4\_TP\_FINAL\_G5\_benchmark.sql

Se usó la función BENCHMARK de sql que repite la búsqueda determinada cantidad de veces (se ingresa como parámetro el número de repeticiones), esto requiere que el select devuelva un solo dato por lo cual tiene que agregarse una función count(), o alguna otra tipo resumen) a la query que uno quiere testear.

Se hicieron 4 comparaciones de consultas óptimas (solo joins) vs subóptimas (subconsultas con joins o where in)



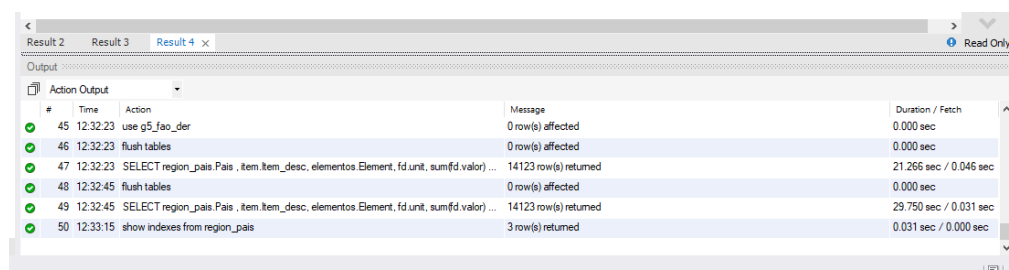
#	Time	Action	Message	Duration / Fetch
9	12:10:30	use g5_fao_der	0 row(s) affected	0.000 sec
10	12:10:30	select * from fao_data	688305 row(s) returned	0.000 sec / 1.218 sec
11	12:10:34	flush tables	0 row(s) affected	0.015 sec
12	12:10:34	SELECT BENCHMARK(1000000, (SELECT count(item_desc) FROM sub_region...	1 row(s) returned	2.657 sec / 0.000 sec
13	12:10:37	flush tables	0 row(s) affected	0.000 sec
14	12:10:37	SELECT BENCHMARK(1000000, (SELECT count(crops_desc) FROM SE...	1 row(s) returned	3.656 sec / 0.000 sec
15	12:10:41	flush tables	0 row(s) affected	0.000 sec
16	12:10:41	SELECT BENCHMARK(2000000, (select count(promedio) from (SELECT ROUND...	1 row(s) returned	2.109 sec / 0.000 sec
17	12:10:43	flush tables	0 row(s) affected	0.000 sec
18	12:10:43	SELECT BENCHMARK(2000000, (SELECT count(datos_promedio) FROM (SELEC...	1 row(s) returned	3.515 sec / 0.000 sec
19	12:10:46	flush tables	0 row(s) affected	0.000 sec
20	12:10:46	SELECT BENCHMARK(3000000, (SELECT count(datos_Element_Code) from (SELEC...	1 row(s) returned	1.968 sec / 0.000 sec
21	12:10:48	flush tables	0 row(s) affected	0.000 sec
22	12:10:48	SELECT BENCHMARK(3000000, (SELECT count(*) from ( Select * from (SELECT fd...	1 row(s) returned	4.703 sec / 0.000 sec
23	12:10:53	flush tables	0 row(s) affected	0.000 sec
24	12:10:53	SELECT BENCHMARK(400000, (SELECT count(*) FROM fao_data AS fd INNER JO...	1 row(s) returned	1.890 sec / 0.000 sec
25	12:10:55	flush tables	0 row(s) affected	0.000 sec
26	12:10:55	SELECT BENCHMARK(400000, (SELECT count(*) from (SELECT fd Pais_Code fd Item...	1 row(s) returned	3.032 sec / 0.000 sec
27	12:10:58	flush tables	0 row(s) affected	0.000 sec

	solo JOINS	subconsultas	Relat al JOINS
Comparación 1	2.657	3.656	38%
Comparación 2	2.109	3.515	67%
Comparación 3	1.969	4.703	139%
Comparación 4	1.890	3.032	60%

Las consultas consideradas subóptimas fueron entre un 38% y 140 % más lentas

### Archivo 5\_TP\_FINAL\_G5\_Benchmark2

Comparación de uso de campos numéricos vs texto indexados en agrupación y ordenamiento (Esto no se pudo realizar con la función benchmark()) por que no permite el ordenamiento ya que al tener que anteponer una función resumen el ordenamiento no tiene sentido)



#	Time	Action	Message	Duration / Fetch
45	12:32:23	use g5_fao_der	0 row(s) affected	0.000 sec
46	12:32:23	flush tables	0 row(s) affected	0.000 sec
47	12:32:23	SELECT region_pais Pais , item.item_desc, elementos Element, fd.unit, sum(fd.valor) ...	14123 row(s) returned	21.266 sec / 0.046 sec
48	12:32:45	flush tables	0 row(s) affected	0.000 sec
49	12:32:45	SELECT region_pais Pais , item.item_desc, elementos Element, fd.unit, sum(fd.valor) ...	14123 row(s) returned	29.750 sec / 0.031 sec
50	12:33:15	show indexes from region_pais	3 row(s) returned	0.031 sec / 0.000 sec

Numérico	Texto	Relat al nro
21.266	29.750	40%

Resulta 40 % más lento agrupar y ordenar por texto

## Archivo 6\_TP\_FINAL\_G5\_prueba\_benchmark.sql

Para comparar mejor la performance con índices o sin ellos se creó una tabla aparte aislada “pruebas” sin índices, y se corre una consulta ordenada por campo numérico (País Code), otra ordenada por campo texto equivalente (País), se crean los índices y se vuelven a correr.

Output

#	Time	Action	Message	Duration / Fetch
✓ 114	12:49:41	flush table	0 row(s) affected	0.000 sec
✓ 115	12:49:41	ALTER TABLE 'g5_fao_der'.pruebas' ADD...	0 row(s) affected Records: 0 Duplicates: 0 ...	3.485 sec
✓ 116	12:49:44	SHOW INDEX FROM pruebas	1 row(s) returned	0.000 sec / 0.000 sec
✓ 117	12:49:44	flush table	0 row(s) affected	0.000 sec
✓ 118	12:49:44	ALTER TABLE 'g5_fao_der'.pruebas' ADD...	0 row(s) affected Records: 0 Duplicates: 0 ...	5.313 sec
✓ 119	12:49:50	SHOW INDEX FROM pruebas	2 row(s) returned	0.000 sec / 0.000 sec
✓ 120	12:49:50	flush table	0 row(s) affected	0.000 sec
✓ 121	12:49:50	select * from pruebas where element_code >...	511844 row(s) returned	1.250 sec / 0.625 sec
✓ 122	12:49:54	flush table	0 row(s) affected	0.015 sec
✓ 123	12:49:54	select * from pruebas where element_code >...	511844 row(s) returned	1.422 sec / 0.594 sec
✓ 124	12:49:58	drop index index_pais_code on pruebas	0 row(s) affected Records: 0 Duplicates: 0 ...	0.031 sec
✓ 125	12:49:58	drop index index_pais on pruebas	0 row(s) affected Records: 0 Duplicates: 0 ...	0.032 sec
✓ 126	12:49:58	SHOW INDEX FROM pruebas	0 row(s) returned	0.000 sec / 0.000 sec
✓ 127	12:49:58	select count(*) from pruebas	1 row(s) returned	0.156 sec / 0.000 sec

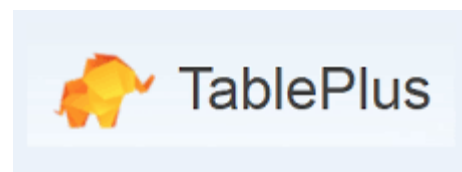
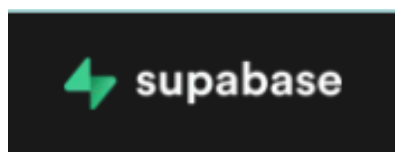
	Número	Texto	Relat al nro
sin indices	1.328	1.438	8%
con índices	1.250	1.422	14%
Relat a sin indices	6%	1%	
creacion indices	3.485	5.313	52%

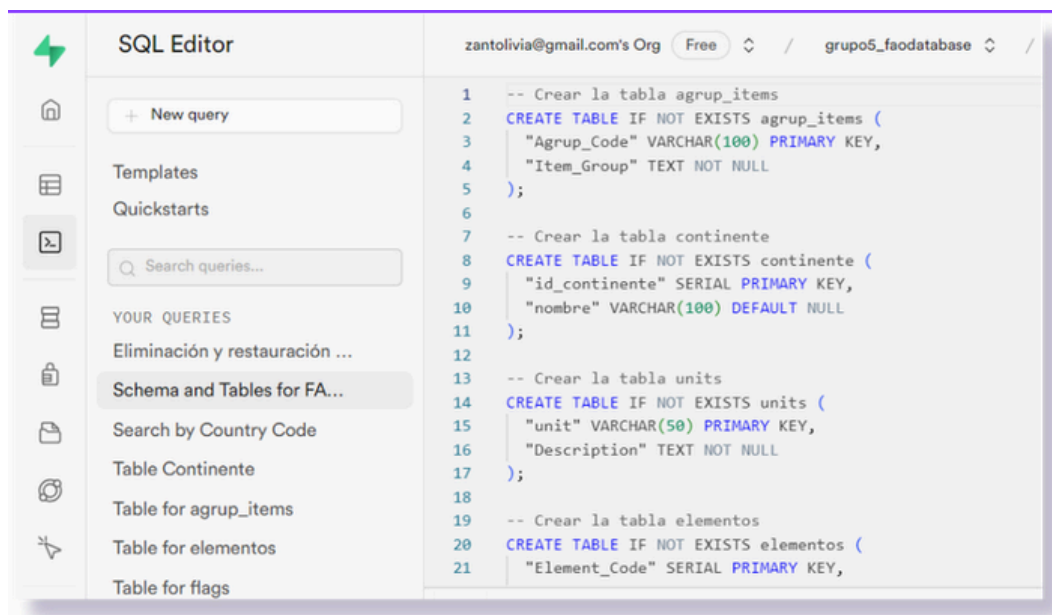
Se observa que los índices mejoran algo la performance no tanto como se esperaría, pero tal vez la poca dispersión de los datos explicaría esto.

Por otro lado se ve que la indexación de campos tipo varchar() resulta bastante más lenta que la de campo numérico.

## Migración de la base en la Nube:

Se usó el servicio backend (con PostgreSQL), y como interfaz gráfica de usuario (GUI) “TablePlus”.

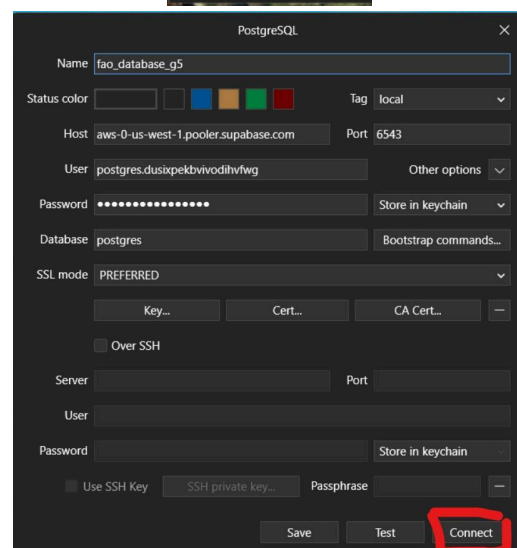
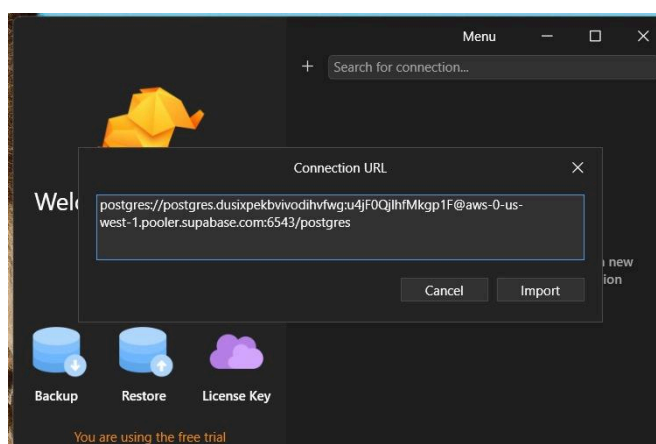
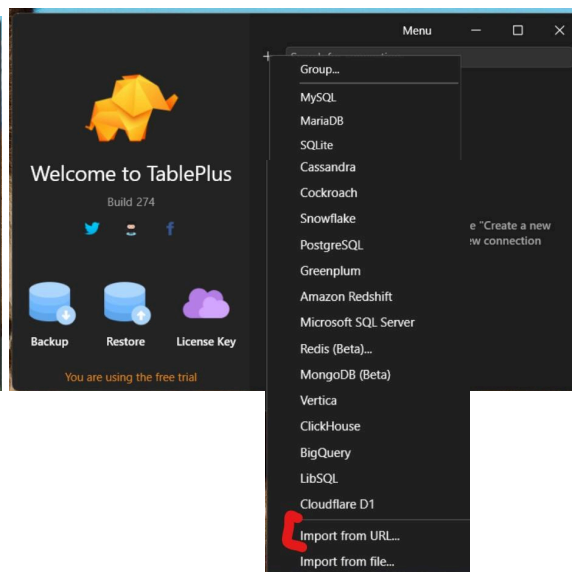
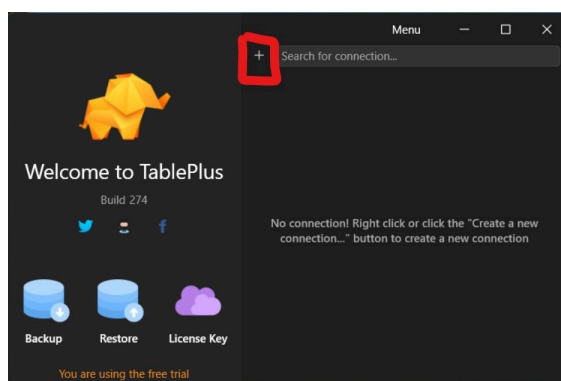




Se repitieron todas las queries y SP/funciones y triggers equivalentes. No se realizó la parte de benchmark dado que no se vio la forma de recuperar el tiempo de las búsquedas.

### Para conectar “Table plus” con “Supabase”

postgres://postgres.dusixpekbviodihvfwg:u4jF0QjIhfMkgp1F@aws-0-us-west-1.pooler.supabase.com:6543/postgres



Agrup_Code	Item_Group
1714	Crops, primary
1717	Cereals, primary
1720	Roots and Tubers, Total
1723	Sugar Crops Primary
1726	Pulses, Total
1729	Treenuts, Total
1730	Oilcrops Primary
1732	Oilcrops, Oil Equivalent
1735	Vegetables Primary
1738	Fruit Primary
1745	Cheese (All Kinds)
1746	Cattle and Buffaloes
1749	Sheep and Goats
1753	Fibre Crops Primary
17530	Fibre Crops, Fibre Equivalent
1756	Live Animals
1765	Meat, Total
1777	Hides and skins, primary
1780	Milk, Total
1783	Eggs Primary
1804	Citrus Fruit, Total
1806	Beef and Buffalo Meat, primary
1807	Chicken and Eggs, primary

Se repitieron todas las queries, SP/funciones y triggers equivalentes. No se realizó la parte de benchmark dado que no se vio la forma de recuperar el tiempo de las búsquedas.  
(archivo 7\_postgreSQL\_G5.sql)

```

1 CREATE OR REPLACE FUNCTION public.pais_inexistente_trigger_func()
2 RETURNS trigger
3 LANGUAGE plpgsql
4 AS $function$
5 DECLARE
6     pais_existente INT;
7     mensaje TEXT;
8 BEGIN
9     -- Verificar si el nuevo Pais_Code existe en la tabla region_pais
10    SELECT COUNT(*) INTO pais_existente
11    FROM region_pais
12    WHERE "region_pais"."Pais_Code" = NEW."Pais_Code";
13
14    -- Si el Pais_Code no existe, generar un mensaje de error
15    IF pais_existente = 0 THEN
16        mensaje := 'Pais_code ' || NEW."Pais_Code" || ' no existente en la tabla region_pais';
17        RAISE EXCEPTION '%', mensaje;
18    END IF;
19
20    RETURN NEW;
21 END;
22 $function$
23

```